Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

2. Übungstest WS 2009

14. Januar 2010

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: Vorname:

Matrikelnummer: Studienkennzahl:

Anzahl abgegebener Zusatzblätter:

Legen Sie bitte Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden.

Die Verwendung von Taschenrechnern, Mobiltelefonen, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

Die Arbeitszeit beträgt 55 Minuten.

	A1:	A2:	A3:	Summe:
Erreichbare Punkte:	24	12	14	50
Erreichte Punkte:				

Viel Erfolg!

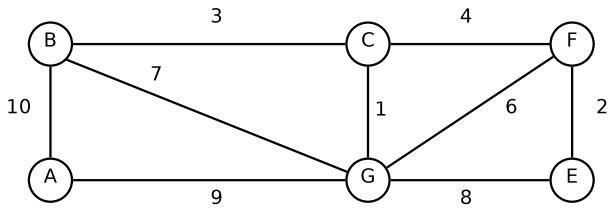
Aufgabe 1.A: Graphen **(24 Punkte)**

a) (14 Punkte)

Gegeben sei ein zusammenhängender, ungerichteter, ungewichteter, schlichter Graph $G = (V, E)$. Schreiben Sie einen effizienten Algorithmus, der den kürzesten Weg (minimale Anzahl an Kanten) zwischen zwei Knoten $x, y \in V$ berechnet und ausgibt. *Hinweis:* Sie können alle in der Vorlesung und in der Übung behandelten abstrakten Datentypen wie Listen, Stacks, Queues, usw. verwenden, *ohne* dass Sie deren Operationen implementieren müssen.

b) (10 Punkte)

Gegeben ist der folgende gewichtete ungerichtete Graph G:



Führen Sie in dem Graphen G den Algorithmus von *Prim* zum Finden eines minimalen Spannbaums durch (die Kantengewichte stehen direkt bei den Kanten) und notieren Sie die genaue Reihenfolge, in der die Kanten in den Baum aufgenommen wurden. *Falls* Sie einen Startknoten benötigen, verwenden Sie den Knoten B.

Die freien Knoten (Knoten die noch nicht in den Baum aufgenommen wurden) werden in einem Minimum-Heap verwaltet. Die Sortierung erfolgt nach dem Gewicht der günstigsten Kante, die einen freien Knoten mit dem aktuellen Baum verbindet. Für alle Knoten, die nicht mit dem aktuellen Baum verbunden werden können, wird ein unendlich großes Gewicht angenommen.

Geben Sie vor dem Einfügen der ersten Kante und dann jeweils nach dem Einfügen einer Kante einen entsprechenden Heap an, der die freien Knoten enthält. *Hinweis:* Eine Beschreibung wie der Heap aktualisiert wird, oder Zwischenschritte sind *nicht* erforderlich, Sie können nach jedem Schritt einen neuen Heap mit den freien Knoten erstellen.

Aufgabe 2.A: Optimierung **(12 Punkte)**

a) (12 Punkte)

Gegeben sei folgende Instanz des 0/1-Rucksackproblems mit Kapazität $K = 5$ und folgenden Gegenständen:

Gegenstand i	Gewicht w_i	Wert c_i
1	2	5
2	6	10
3	2	4
4	2	2
5	1	4
6	3	5
7	1	2

$i\backslash j$	0	1	2	3	4	5
0	0	0	0	0	0	0
1						
2						
3						
4						
5						
6						
7						

Diese Instanz soll mittels *Dynamischer Programmierung* über die möglichen Gesamtgewichte gelöst werden. Dazu wird die oben stehende 8×6-Matrix $\mathbf{m}$ verwendet, wobei der Eintrag im Feld $m_{i,j}$ angibt, welcher Wert mit den ersten i Gegenständen erreicht werden kann, wenn das Gesamtgewicht der gewählten Gegenstände kleiner oder gleich j ist.

Die Felder der Matrix können wie folgt berechnet werden:

$$m_{0,j} = 0 \qquad \text{für } j = 0, \ldots, 5$$

$$m_{i,j} = \begin{cases} m_{i-1,j} & \text{falls } w_i > j, \\ \max\{m_{i-1,j-w_i} + c_i,\ m_{i-1,j}\} & \text{sonst.} \end{cases} \qquad \text{für } \begin{cases} i = 1, \ldots, 7 \\ j = 0, \ldots, 5 \end{cases}$$

- Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.

- Geben Sie eine optimale Lösung der Probleminstanz an, und markieren Sie in der Matrix jene Zellen, die beim Rekonstruieren der Lösung betrachtet werden.

- Welche Laufzeit ergibt sich bei dieser Rekonstruktion in Θ-Notation in Abhängigkeit der Anzahl der Elemente n und der Kapazität K?

Aufgabe 3.A: Hash und Suchverfahren **(14 Punkte)**

a) (12 Punkte)

Gegeben seien Hashtabellen mit Tabellengröße $m = 7$, die zur Kollisionsbehandlung Double Hashing verwendet. *Hinweis: Beim Einfügen muss für jedes Element der Tabelle eindeutig ersichtlich sein, wie es an seinen Platz kommt.*

$$h_1(k) = k \bmod 7$$
$$h_2(k) = (k \bmod 6) - 7$$

- Fügen Sie den Wert 10 **mit** der Verbesserung nach Brent in die folgende Tabelle ein:

0	1	2	3	4	5	6
7			3			13

- Fügen Sie den Wert 8 **mit** der Verbesserung nach Brent in die folgende Tabelle ein:

0	1	2	3	4	5	6
	1	2	3		5	

- Fügen Sie den Wert 8 **ohne** der Verbesserung nach Brent in die folgende Tabelle ein:

0	1	2	3	4	5	6
	1	2	3		5	

- Welches Problem kann beim Einfügen in die Hashtabelle bei der Verwendung der oben angegebenen Hashfunktionen auftreten? Wie müssen die Funktionen $h_1(k)$ und/oder $h_2(k)$ abgeändert werden, um dieses Problem zu beheben?

b) (2 Punkte)

Welche Vor- bzw. Nachteile hat die Verkettung der Überläufer im Vergleich zu offenen Hashverfahren?

Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

2. Übungstest WS 2009

14. Januar 2010

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: Vorname:

Matrikelnummer: Studienkennzahl:

Anzahl abgegebener Zusatzblätter:

Legen Sie bitte Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden.

Die Verwendung von Taschenrechnern, Mobiltelefonen, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

Die Arbeitszeit beträgt 55 Minuten.

	B1:	B2:	B3:	Summe:
Erreichbare Punkte:	14	24	12	50
Erreichte Punkte:				

Viel Glück!

Aufgabe 1.B: Hash und Suchverfahren **(14 Punkte)**

a) (2 Punkte)

Beschreiben Sie kurz die Funktionsweise der Multiplikationsmethode und geben sie eine Hashfunktion $h(k)$ an, die auf dieser Methode beruht.

b) (12 Punkte)

Gegeben seien Hashtabellen mit Tabellengröße $m = 7$, die zur Kollisionsbehandlung Double Hashing verwendet. *Hinweis: Beim Einfügen muss für jedes Element der Tabelle eindeutig ersichtlich sein, wie es an seinen Platz kommt.*

$$h_1(k) = k \bmod 7$$
$$h_2(k) = (k \bmod 6) - 7$$

- Fügen Sie den Wert 11 **mit** der Verbesserung nach Brent in die folgende Tabelle ein:

0	1	2	3	4	5	6
	1	2		4		

- Fügen Sie den Wert 4 **ohne** der Verbesserung nach Brent in die folgende Tabelle ein:

0	1	2	3	4	5	6
	8	2		11	5	

- Fügen Sie den Wert 4 **mit** der Verbesserung nach Brent in die folgende Tabelle ein:

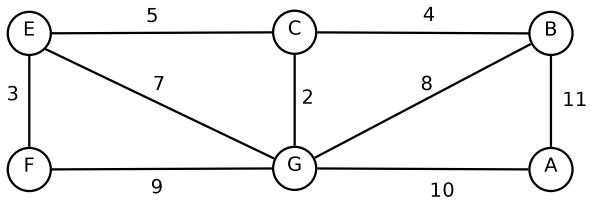
0	1	2	3	4	5	6
	8	2		11	5	

- Welches Problem kann beim Einfügen in die Hashtabelle bei der Verwendung der oben angegebenen Hashfunktionen auftreten? Wie müssen die Funktionen $h_1(k)$ und/oder $h_2(k)$ abgeändert werden, um dieses Problem zu beheben?

Aufgabe 2.B: Graphen **(24 Punkte)**

a) (10 Punkte)

Gegeben ist der folgende gewichtete ungerichtete Graph G:



Führen Sie in dem Graphen G den Algorithmus von *Prim* zum Finden eines minimalen Spannbaums durch (die Kantengewichte stehen direkt bei den Kanten) und notieren Sie die genaue Reihenfolge, in der die Kanten in den Baum aufgenommen wurden. *Falls* Sie einen Startknoten benötigen, verwenden Sie den Knoten B.

Die freien Knoten (Knoten die noch nicht in den Baum aufgenommen wurden) werden in einem Minimum-Heap verwaltet. Die Sortierung erfolgt nach dem Gewicht der günstigsten Kante, die einen freien Knoten mit dem aktuellen Baum verbindet. Für alle Knoten, die nicht mit dem aktuellen Baum verbunden werden können, wird ein unendlich großes Gewicht angenommen.

Geben Sie vor dem Einfügen der ersten Kante und dann jeweils nach dem Einfügen einer Kante einen entsprechenden Heap an, der die freien Knoten enthält. *Hinweis:* Eine Beschreibung wie der Heap aktualisiert wird, oder Zwischenschritte sind *nicht* erforderlich, Sie können nach jedem Schritt einen neuen Heap mit den freien Knoten erstellen.

b) (14 Punkte)

Gegeben sei ein zusammenhängender, ungerichteter, ungewichteter, schlichter Graph $G = (V, E)$. Schreiben Sie einen effizienten Algorithmus, der den kürzesten Weg (minimale Anzahl an Kanten) zwischen zwei Knoten $x, y \in V$ berechnet und ausgibt. *Hinweis:* Sie können alle in der Vorlesung und in der Übung behandelten abstrakten Datentypen wie Listen, Stacks, Queues, usw. verwenden, *ohne* dass Sie deren Operationen implementieren müssen.

Aufgabe 3.B: Optimierung **(12 Punkte)**

a) (12 Punkte)

Gegeben sei folgende Instanz des 0/1-Rucksackproblems mit Kapazität $K = 5$ und folgenden Gegenständen:

Gegenstand i	Gewicht w_i	Wert c_i
1	2	5
2	2	4
3	6	10
4	1	4
5	2	2
6	3	5
7	1	2

$i \backslash j$	0	1	2	3	4	5
0	0	0	0	0	0	0
1						
2						
3						
4						
5						
6						
7						

Diese Instanz soll mittels *Dynamischer Programmierung* über die möglichen Gesamtgewichte gelöst werden. Dazu wird die oben stehende 8×6-Matrix $\mathbf{m}$ verwendet, wobei der Eintrag im Feld $m_{i,j}$ angibt, welcher Wert mit den ersten i Gegenständen erreicht werden kann, wenn das Gesamtgewicht der gewählten Gegenstände kleiner oder gleich j ist.

Die Felder der Matrix können wie folgt berechnet werden:

$$m_{0,j} = 0 \qquad \text{für } j = 0, \ldots, 5$$

$$m_{i,j} = \begin{cases} m_{i-1,j} & \text{falls } w_i > j, \\ \max\{m_{i-1,j-w_i} + c_i,\ m_{i-1,j}\} & \text{sonst.} \end{cases} \qquad \text{für } \begin{cases} i = 1, \ldots, 7 \\ j = 0, \ldots, 5 \end{cases}$$

- Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.

- Geben Sie eine optimale Lösung der Probleminstanz an, und markieren Sie in der Matrix jene Zellen, die beim Rekonstruieren der Lösung betrachtet werden.

- Welche Laufzeit ergibt sich bei dieser Rekonstruktion in Θ-Notation in Abhängigkeit der Anzahl der Elemente n und der Kapazität K?